Transfer Learning in Collaborative Recommendation for Bias Reduction

Zinan Lin
College of Computer Science and Software Engineering, Shenzhen University
Shenzhen, China
lzn87591@gmail.com

Dugang Liu
College of Computer Science and Software Engineering, Shenzhen University
Shenzhen, China
dugang.ldg@gmail.com

Weike Pan*
Zhong Ming*
Shenzhen University
Shenzhen, China
panweike,mingz@szu.edu.cn

ABSTRACT

In a recommender system, a user's interaction is often biased by the items' displaying positions and popularity, as well as the user's self-selection. Most existing recommendation models are built using such a biased user-system interaction data. In this paper, we first additionally introduce a specially collected unbiased data and then propose a novel transfer learning solution, i.e., transfer via joint reconstruction (TJR), to achieve knowledge transfer and sharing between the biased data and unbiased data. Specifically, in our TJR, we refine the prediction via the latent features containing bias information in order to obtain a more accurate and unbiased prediction. Moreover, we integrate the two data by reconstructing their interaction in a joint learning manner. We then adopt three representative methods as the backbone models of our TJR and conduct extensive empirical studies on two public datasets, showcasing the effectiveness of our transfer learning solution over some very competitive baselines.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Transfer Learning, Collaborative Recommendation, Bias Reduction, Unbiased Data

ACM Reference Format:
Zinan Lin, Dugang Liu, Weike Pan, and Zhong Ming. 2021. Transfer Learning in Collaborative Recommendation for Bias Reduction. In *Fifteenth ACM Conference on Recommender Systems (RecSys '21), September 27-October 1, 2021, Amsterdam, Netherlands.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3460231.3478860

1 INTORDUCTION

In a typical closed-loop recommender system, a user's feedback or interaction is often influenced by the items' displaying positions and popularity, etc, which means that the collected data are biased [15, 16]. Most current collaborative recommendation models are built with the biased data only, which may not suit the users' tastes well. In this paper, we turn to leverage an additional unbiased data for users' preference learning, which is thus called collaborative recommendation with biased and unbiased data.

For the studied recommendation problem with both biased and unbiased data, previous works show that the unbiased data has the potential to mitigate the bias in users' feedback [8, 10, 16]. However, the unbiased data is usually rather small due to the high expensiveness of collection in a deployed online system, which makes it difficult to help reduce the bias in the relatively big biased data. Some researchers have made some pioneering efforts in this direction, including proposing a domain adaptation algorithm [1], developing a knowledge-distillation framework [8], and designing a meta learning algorithm [2], etc. However, these methods do not fully consider the difference between the biased and unbiased data in the generation process, which may not address the bias challenge in the biased data and the heterogeneity challenge of the two different data well.

As a response, in this paper, we propose a novel transfer learning solution called transfer via joint reconstruction (TJR). Specifically, we first convert the modeling of the biased and unbiased data into a transfer learning problem, where the big biased data is taken as the auxiliary data and the small unbiased data is taken as the target data. We extract the latent features that represent users' preferences and bias information from two different data. Intuitively, regardless of what the latent features represent, they ultimately affect the prediction of the model. Therefore, we refine the prediction by the latent features containing bias information to obtain a more accurate and unbiased prediction, which thus alleviates the bias problem. In order to effectively achieve knowledge transfer between the auxiliary data and the target data for bias reduction, we propose a joint reconstruction loss, aiming to guide the reconstructed output to fit both the target data and auxiliary data well. In this way, the unbiased target data implicitly plays a role of high-quality data that guides the learning task of the biased data, which addresses both the bias challenge and the heterogeneity challenge.

2 RELATED WORK

2.1 Collaborative Recommendation

In collaborative recommendation for a user u, we aim to find the most relevant or interested items from a candidate pool, which is usually achieved by learning a preference score between a user u and each item i, i.e., $\hat{r}_{ui}$. In order to estimate $\hat{r}_{ui}$, we may use a neighborhood-based method [3], a factorization-based method [5, 12] or a deep network based method [4, 7]. A neighborhood-based

*co-corresponding authors

RecSys '21, September 27-October 1, 2021, Amsterdam, Netherlands

ACM ISBN 978-1-4503-8458-2/21/09.
https://doi.org/10.1145/3460231.3478860

method first calculates the similarities among the users (or items) and then predicts the preference $\hat{r}_{ui}$ by aggregating the preferences of the neighbors of user u to item i (or the preferences of user u to the neighbors of item i). A factorization-based method often factorizes the user-item interaction matrix to learn the latent representations of the users and items. A deep neural network based method is able to capture the non-linear interaction between the latent features of users and items.

2.2 Bias Reduction in Collaborative Filtering

In recent years, the bias problem in collaborative recommendation has gradually received more attention by the researchers and practitioners from the academia and industry. Existing works for bias reduction can be categorized into three classes, including counterfactual learning-based methods [14, 15, 17], heuristic-based methods [9] and unbiased data augmentation methods [1, 2, 8]. One of the most popular counterfactual learning-based methods is inverse propensity score (IPS) [6], where each sample is assigned a weight via a theoretically unbiased prediction model. Heuristic-based methods are relatively few, which often make certain assumptions about the data being missing not at random. The data augmentation methods alleviate the bias problem by introducing an unbiased data collected by a specific random policy, which is recognized as a very promising approach.

Our TJR aims to transfer knowledge between a biased data and an unbiased data for joint preference modeling, and reduce the bias via a specially designed prediction refinement component.

3 OUR SOLUTION: TRANSFER VIA JOINT RECONSTRUCTION

3.1 Problem Definition

In our studied problem of collaborative recommendation with biased and unbiased data, we have a set of users and a set of items, denoted by $\mathcal{U} = \{u\} = \{1, 2, ..., n\}$ and $\mathcal{I} = \{i\} = \{1, 2, ..., m\}$, respectively. Moreover, we have two different sets of user-item interaction feedback, i.e., $S^{\mathcal{A}} = \{(u, i)\}$ and $S^{\mathcal{T}} = \{(u, i)\}$ obtained by two different policies. Specifically, $S^{\mathcal{A}}$ is a big and biased data collected by a commonly deployed non-uniform policy in a typical online recommender system and $S^{\mathcal{T}}$ is a small and unbiased data collected by a specially designed uniform (i.e., random) policy. Our goal is then to reduce the bias in $S^{\mathcal{A}}$ and provide a more accurate personalized recommendation list of items for each user.

3.2 Our Solution: Transfer via Joint Reconstruction

We study the problem from the perspective of transfer learning [13]. We regard $S^{\mathcal{A}}$ and $S^{\mathcal{T}}$ as the auxiliary data and the target data, respectively. From the idea of latent variable model, we can learn users' latent features from $S^{\mathcal{A}}$ via sufficient training. However, due to the existence of bias, these latent features are mixed with features containing bias information. Meanwhile, we can not obtain users' unbiased latent features well from $S^{\mathcal{T}}$ because its scale is often too small.

In order to make full use of the information in $S^{\mathcal{A}}$ and $S^{\mathcal{T}}$, we propose a novel transfer learning solution, i.e., transfer via joint reconstruction (TJR), which is illustrated in Figure 1. Intuitively, the learned latent features, whether they represent users' preferences or bias information, ultimately affect the prediction of the model. Therefore, we use two different models to extract the latent features that represent users' preferences and bias information, and then refine the prediction in a linear manner to alleviate bias problem.

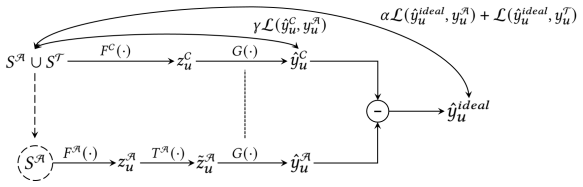


Figure 1: Illustration of our transfer via joint reconstruction (TJR). We extract the corresponding information from two different data. $F^{\mathcal{C}}(\cdot)$ is used to extract confusing latent features $z_u^{\mathcal{C}}$ that contain both user preferences and bias information, and both $F^{\mathcal{A}}(\cdot)$ and $T^{\mathcal{A}}(\cdot)$ are used to extract latent features $z_u^{\mathcal{A}}$ about bias information. Based on $z_u^{\mathcal{C}}$, $z_u^{\mathcal{A}}$ and $G(\cdot)$, we can get the unbiased prediction $\hat{y}_u^{ideal}$. Notice that $G(\cdot)$ is shared and the arcs denote the loss functions to be minimized simultaneously, i.e., reconstructing both biased and unbiased data, to achieve the effect of joint reconstruction.

Specifically, we use the union of $S^{\mathcal{A}}$ and $S^{\mathcal{T}}$ to obtain the confusing latent features $z_u^{\mathcal{C}}$ that contain user preferences and bias information through the function for latent feature extraction, i.e., $F^{\mathcal{C}}(\cdot)$. After getting $z_u^{\mathcal{C}}$, we use $G(\cdot)$ to get the prediction $\hat{y}_u^{\mathcal{C}}$. Notice that $\hat{y}_u^{\mathcal{C}}$ is usually biased. In particular, if we use VAE as the backbone model, $F^{\mathcal{C}}(\cdot)$ refers to the encoder and $G(\cdot)$ refers to the decoder.

We use $S^{\mathcal{A}}$ to extract latent features of bias information $\tilde{z}_u^{\mathcal{A}}$ through $F^{\mathcal{A}}(\cdot)$ and $T^{\mathcal{A}}(\cdot)$. Notice that $F^{\mathcal{A}}(\cdot)$ is the same as $F^{\mathcal{C}}(\cdot)$ in structure and $T^{\mathcal{A}}(\cdot)$ is a transform function. The reason for introducing the transform function is that the latent features of bias are usually non-linear and high-dimensional. We use sigmoid as the default transform function in our TJR and also study the impact of different ones on the recommendation performance. After obtaining $\tilde{z}_u^{\mathcal{A}}$, we use $G(\cdot)$ to get the prediction $\hat{y}_u^{\mathcal{A}}$. Notice that $G(\cdot)$ in the bottom branch is shared with the the one in the upper branch that generates $\hat{y}_u^{\mathcal{C}}$. The reason is that $G(\cdot)$ serves to map the values in the latent feature space to the label space, and the mapping relationship is theoretically unchanging regardless of whether the latent features are biased or unbiased.

To obtain an unbiased prediction $\hat{y}_u^{ideal}$, we intuitively use a linear method through $\hat{y}_u^{\mathcal{C}}$ and $\hat{y}_u^{\mathcal{A}}$, i.e., $\hat{y}_u^{ideal} = \hat{y}_u^{\mathcal{C}} - \hat{y}_u^{\mathcal{A}}$. Finally, in order to train our TJR, we naturally design a joint reconstruction loss function, i.e., reconstructing both the biased and unbiased data. For user u, we can easily obtain the loss function, i.e., $\alpha\mathcal{L}(\hat{y}_u^{ideal}, y_u^{\mathcal{A}}) + \mathcal{L}(\hat{y}_u^{ideal}, y_u^{\mathcal{T}})$, where $\mathcal{L}(\cdot, \cdot)$ denotes an arbitrary loss function such as the cross-entropy loss. Notice that we follow the idea of the Weight strategy [8] and introduce a hyper-parameter α in the above loss function. To learn the biased features $z_u^{\mathcal{C}}$ better, we introduce an additional loss function, i.e., $\mathcal{L}(\hat{y}_u^{\mathcal{C}}, y_u^{\mathcal{A}})$. Finally, we

have the overall loss function of our TJR,

$$\mathcal{L}_{\text{TJR}} = \alpha \mathcal{L}(\hat{y}_u^{ideal}, y_u^{\mathcal{A}}) + \mathcal{L}(\hat{y}_u^{ideal}, y_u^{\mathcal{T}}) + \gamma \mathcal{L}(\hat{y}_u^{C}, y_u^{\mathcal{A}}), \quad (1)$$

where α and γ are the hyper-parameters, $y_u^{\mathcal{A}}$ are the observed labels from $S_u^{\mathcal{A}}(S_u^{\mathcal{A}} \subseteq S^{\mathcal{A}})$ and $y_u^{\mathcal{T}}$ from $S_u^{\mathcal{T}}(S_u^{\mathcal{T}} \subseteq S^{\mathcal{T}})$.

4 EXPERIMENTS

4.1 Experimental Setup

Datesets. For the studied problem of collaborative recommendation with biased data and unbiased data, there are only two public datasets available, i.e., Yahoo! R3 [11] and Coat Shopping [15], which are thus both included in our experiments. Yahoo! R3 is a (user, song) rating data with a biased user subset and an unbiased random subset, involving 15400 users and 1000 songs. The user subset is biased, collected under a common recommendation policy and the random subset is an unbiased data, collected under a uniform policy. We follow a previous work [8], and regard the ratings larger than 3 as positive feedback. In addition, we regard the biased user subset as the auxiliary data ($S^{\mathcal{A}}$) and randomly split the unbiased random subset into three subsets: 10% as the target data ($S^{\mathcal{T}}$) for training, 10% as the validation data (S_{va}) to tune the hyper-parameters, and the rest 80% as the test data (S_{te}) for performance evaluation. Coat Shopping is a (user, coat) rating data collected from 290 Amazon Mechanical Turk (AMT) workers on an inventory of 300 coats. Similar to Yahoo! R3, it contains a biased user subset and an unbiased random subset. The preprocessing and split of the two subsets are the same as that on Yahoo! R3.

Metrics. For evaluation of collaborative recommendation, we adopt two widely used metrics in the community of recommender systems, including the area under the ROC curve (AUC) and normalized discounted cumulative gain (NDCG@50), where AUC is the main metric.

Baselines. We use three representative methods as backbone models (BMs), i.e., variational autoencoders (VAE), matrix factorization (MF) and neural collaborative filtering (NCF). Notice that for MF and NCF, we use $S^{\mathcal{A}} \cup S^{\mathcal{T}}$ instead of $S^{\mathcal{A}}$ for learning the bias information in order to avoid the situation when a sampled triple is included in $S^{\mathcal{A}} \cup S^{\mathcal{T}}$ but not in $S^{\mathcal{A}}$, because the latent features are learned by randomly sampling one single (user, item, rating) triple rather than all the rating records of a user in VAE. Moreover, for the unobserved (user, item) pairs, when VAE is used as the backbone model, we treat them as negative feedback, and when MF and NCF are used as the backbone models, we treat them as missing values.

Our TJR is closely related with the backbone models, and thus use BM($S^{\mathcal{A}}$), BM($S^{\mathcal{T}}$) and BM($S^{\mathcal{A}} \cup S^{\mathcal{T}}$) to denote the used data sources, i.e., training only with $S^{\mathcal{A}}$, training only with $S^{\mathcal{T}}$, and training with both $S^{\mathcal{A}}$ and $S^{\mathcal{T}}$, respectively, where BM is a specific backbone model. We also compare the representative methods proposed in recent years, including inverse propensity score (IPS) [15] and CausE [1], as well as the Bridge and Weight strategies [8]. For IPS, since we use implicit feedback instead of ratings, we estimate the propensity score w.r.t. an item i via the naive Bayes estimator that considers the exposure of the item [6, 14]. In addition, a model called AutoDebias has recently been proposed [2], which is also included as one of the baselines. However, AutoDebias only uses matrix factorization as the backbone model, and it is difficult to be extended to VAE and NCF. Hence, we do not compare with AutoDebias using VAE or NCF as the backbone model.

Reproducibility. For AutoDebias-MF, we implement it via PyTorch 1.1 using the MSE loss and parameter configurations following the original paper [2]. For all the other methods, we implement them via TensorFlow 1.2 using the cross-entropy loss and batch training. We conduct a grid search to tune the hyper-parameters of all the methods by checking the AUC performance on the validation data S_{va}. Specifically, we choose the embedding size $rank \in \{50, 100, 200\}$, the hyper-parameter on the regularization term $\lambda \in \{1e^{-5}, 1e^{-4}, \ldots, 1\}$, and the tradeoff hyper-parameters $\alpha \in \{0.1, 0.2, \ldots, 1.0\}$ and $\gamma \in \{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0\}$. For the methods using VAE as the backbone model, we use RMSProp as the optimizer (the learning rate is fixed as 0.0001), set the weight on the KL divergence as 0.2, fix the iteration number as 300, and choose the dropout rate from $\{0.2, 0.3, 0.4, 0.5\}$. For the methods using MF (except AutoDebias-MF) and NCF as the backbone models, we use Adam as the optimizer (the learning rate is fixed as 0.001), and set the iteration number as 100. Notice that we adopt an early stopping strategy with the patience set to 5 times for the methods using NCF as the backbone model. In addition, due to the relatively small size of the validation data, the selection of the optimal values is not always very stable. Therefore, we run ten times, and then select the one with the best average performance. Notice that the source code and scripts to reproduce all the results are publicly available at https://csse.szu.edu.cn/staff/panwk/publications/TJR/.

4.2 RQ1: Performance Comparison

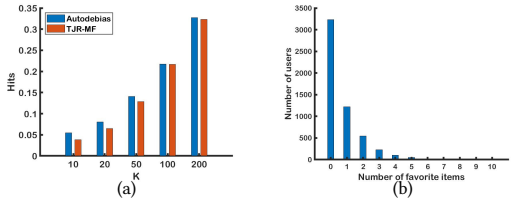
We report the main results in Table 1, from which we can have the following observations: (i) Overall, our TJR outperforms all the baselines on both datasets, which clearly shows the advantage of our transfer learning solution in jointly modeling the biased and unbiased data. Notice that we take AUC as the main metric and use it in the process of tuning the hyper-parameters. The loss function of our TJR is similar to that of the Weight strategy [8], but the performance of our model are better, which further illustrates the validity of our TJR. (ii) Our TJR is flexible and could be easily extended to different backbone models such as MF, VAE and NCF. (iii) The performance of both CausE and the Bridge strategy are limited by the pre-trained model obtained by $S^{\mathcal{T}}$, and are also limited by the scale of $S^{\mathcal{T}}$. Notice that our TJR jointly reconstructs $S^{\mathcal{A}}$ and $S^{\mathcal{T}}$, which alleviates the problem of small size of $S^{\mathcal{T}}$ to a certain extent. (iv) In the experiments with matrix factorization as the backbone model, the performance of our TJR is worse than AutoDebias on NDCG@50 on Yahoo! R3. We analyse the prediction of AutoDebias and our TJR-MF, and find that the hits of the two models are not very different and most users' hits are 0s as shown in Figure 2, which is caused by the nature of the dataset. Specifically, in the test set, the number of positive feedback of each user is rather small. This situation is very different from most recommendation settings on public biased datasets in previous works. Therefore, for bias reduction, using NDCG as an evaluation metric may not be the best, which may only be used as an auxiliary metric.

Table 1: Recommendation performance on Yahoo! R3 and Coat Shopping, where the best results are marked in bold and the second best results are marked in underline. Notice that we follow the original paper and only use matrix factorization as the backbone model in AutoDebias, since it is more difficult to support other backbone models such as VAE and NCF.

Dataset	Metrics	VAE($S^{\mathcal{A}}$)	VAE($S^{\mathcal{T}}$)	VAE($S^{\mathcal{A}} \cup S^{\mathcal{T}}$)	IPS-VAE	CausE-VAE	Bridge-VAE	Weight-VAE	AutoDebias-VAE	TJR-VAE
Yahoo! R3	AUC	0.7666	0.5770	0.7709	0.7470	0.7673	0.7711	0.7723	-	**0.7804**
	NDCG@50	0.1009	0.0258	0.1014	0.0809	0.1013	0.1007	0.0998	-	**0.1023**
Coat Shopping	AUC	0.6210	0.5413	0.6269	0.5757	0.6210	0.6210	0.6245	-	**0.7603**
	NDCG@50	0.0921	0.0807	0.0952	0.0856	0.0922	0.0932	0.0949	-	**0.1239**

Dataset	Metrics	MF($S^{\mathcal{A}}$)	MF($S^{\mathcal{T}}$)	MF($S^{\mathcal{A}} \cup S^{\mathcal{T}}$)	IPS-MF	CausE-MF	Bridge-MF	Weight-MF	AutoDebias	TJR-MF
Yahoo! R3	AUC	0.7329	0.5684	0.7409	0.7346	0.7285	0.7524	0.7465	0.7472	**0.7696**
	NDCG@50	0.0382	0.0304	0.0439	0.0426	0.0445	0.0615	0.0494	**0.0870**	0.0705
Coat Shopping	AUC	0.7606	0.5231	0.7631	0.7636	0.7611	**0.7653**	0.7636	0.6965	0.7646
	NDCG@50	0.0990	0.0578	0.1016	0.0965	0.0985	0.1005	0.1012	0.0999	**0.1027**

Dataset	Metrics	NCF($S^{\mathcal{A}}$)	NCF($S^{\mathcal{T}}$)	NCF($S^{\mathcal{A}} \cup S^{\mathcal{T}}$)	IPS-NCF	CausE-NCF	Bridge-NCF	Weight-NCF	AutoDebias-NCF	TJR-NCF
Yahoo! R3	AUC	0.7245	0.6050	0.7268	0.7273	0.7283	0.7367	0.7380	-	**0.7420**
	NDCG@50	0.0279	0.0275	0.0327	0.0304	0.0284	0.0439	0.0383	-	**0.0454**
Coat Shopping	AUC	0.7507	0.5840	0.7508	0.7337	0.7516	0.7522	0.7509	-	**0.7537**
	NDCG@50	0.0976	0.0781	0.0969	0.0902	0.0961	0.0969	0.0946	-	**0.0995**



(a) (b)

Figure 2: The percentage of hits of AutoDebias and our TJR-MF (a), and the distribution of the number of users over the number of favorite items in the test set (b), for Yahoo! R3.

4.3 RQ2: Ablation Studies

In order to gain some deep understanding of our TJR, we use VAE as the backbone model and conduct some ablation studies by removing some components from the framework of our TJR, and report the results in Figure 3(a-b). Firstly, we cut off the sharing path of $G(\cdot)$, denoted as "-Share". We can see that sharing $G(\cdot)$ can improve the performance of our model because that could force the users' latent feature space and the bias's latent feature space to be as close as possible, which is beneficial for our TJR to alleviate the bias problem from the same angle. In addition, we remove the branch that extracts latent features of bias information so that our TJR degenerates into VAE($S^{\mathcal{A}} \cup S^{\mathcal{T}}$), denoted as "-Sub". We can see that the recommendation performance of our TJR without the bias branch degrades significantly, which shows the usefulness of the designed bias reduction component.

4.4 RQ3: Impact of the Transform Function

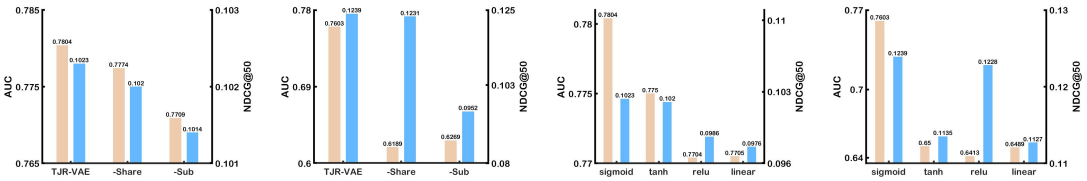
In this subsection, we again use VAE as the backbone model and further study the impact of the transform function used to extract the bias information. Specifically, we report the the recommendation performance of our TJR with different transform functions including sigmoid, tanh, relu and linear, which are shown in Figure 3(c-d). We can see that using a nonlinear activation function improves the performance of our TJR more significantly than a linear one. Among them, using the sigmoid function performs the best, followed by using tanh. The reason is related to the property of the bias itself. Notice that the performance of a recommendation model largely depends on whether it learns the users' interests or not, and the impact of bias on the performance is not very significant. In other words, the value of the bias features could not be too large to ensure the overall performance of our TJR via the sigmoid function.

4.5 RQ4: Impact of the Hyper-Parameters

In this subsection, we use VAE as the backbone model and study the impact of the hyper-parameters α and γ in our TJR, and show the results in Figure 4. Specifically, we first fix γ as the optimal value, and report the recommendation performance with different values of $\alpha \in \{0.1, 0.2, ..., 1.0\}$. We can see that the best values on Yahoo! R3 and Coat Shopping are 0.2 and 0.6, respectively. The reason is that the ratio $|S^{\mathcal{A}}|/|S^{\mathcal{T}}|$ of Yahoo! R3 is larger than that of Coat Shopping. In order to reduce the effect of the loss on $S^{\mathcal{A}}$, α shall be smaller. Similarly, we fix α as the optimal value, and report the recommendation performance with different values of $\gamma \in \{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0\}$. We can see that the best values on Yahoo! R3 and Coat Shopping are both 0.5. For user u, $\mathcal{L}(\hat{y}_u^C, y_u^{\mathcal{A}})$ is used to learn the biased features z_u^C better and the role of γ is to control the its proportion. If γ is too small, the users' latent features may not be fully trained, and if γ is too large, the effect of the bias branch in TJR will be weakened.

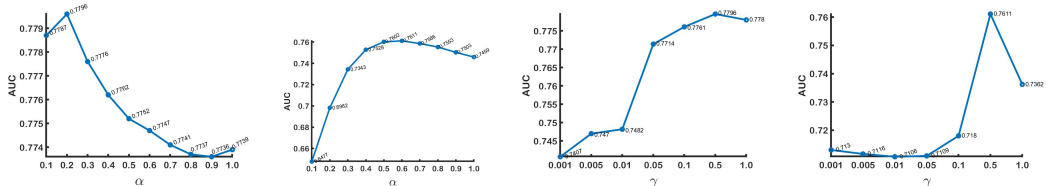
5 CONCLUSIONS AND FUTURE WORK

In this paper, we study an emerging and important problem called collaborative recommendation with a big biased data and a small unbiased data. As a response, we view this problem from a transfer learning perspective, and propose a novel transfer learning solution to achieve knowledge transfer between the two different data, aiming to reduce the bias and improve the recommendation performance. Specifically, we design an end-to-end transfer learning framework, including two different but related models to extract latent features that represent users' preferences and bias information,



(a) Yahoo! R3 (b) Coat (c) Yahoo! R3 (d) Coat

Figure 3: Recommendation performance of our TJR by removing different components (i.e., "-Share" and "-Sub"), and using different transform functions (i.e., sigmoid, tanh, relu and linear), which are shown in (a-b) and (c-d), respectively. Notice that "-Share" and "-Sub" denotes removing the sharing path of $G(\cdot)$ and removing the branch used to extract the bias information, respectively.



(a) Yahoo! R3 (b) Coat Shopping (c) Yahoo! R3 (d) Coat Shopping

Figure 4: Recommendation performance of our TJR on Yahoo! R3 and Coat Shopping with different values of $\alpha \in \{0.1, 0.2, ..., 1.0\}$ and $\gamma \in \{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0\}$, which are shown in (a-b) and (c-d), respectively.

a bias reduction component and a shared prediction model, optimized by a joint reconstruction loss, which is thus called transfer via joint reconstruction (TJR). We then conduct extensive empirical studies on two public datasets, and find that our TJR performs significantly better than some very competitive baseline methods in most cases.

For future works, we are interested in further generalizing our transfer learning solution to include more information such as temporal dynamics and item descriptions.

ACKNOWLEDGMENTS

We thank the support of National Natural Science Foundation of China Nos. 61836005 and 61872249, and Shenzhen Basic Research Fund No. JCYJ20200813091134001.

REFERENCES

[1] Stephen Bonner and Flavian Vasile. 2018. Causal embeddings for recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*. 104–112.
[2] Jiawei Chen, Hande Dong, Yang Qiu, Xiangnan He, Xin Xin, Liang Chen, Guli Lin, and Keping Yang. 2021. AutoDebias: Learning to debias for recommendation. *arXiv preprint arXiv:2105.04170* (2021).
[3] Mukund Deshpande and George Karypis. 2004. Item-based top-N recommendation algorithms. *ACM Transactions on Information Systems* 22, 1 (2004), 143–177.
[4] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. 173–182.
[5] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.
[6] Dawen Liang, Laurent Charlin, and David M Blei. 2016. Causal inference for recommendation. In *Workshop on Causation: Foundation to Application co-located with the 32nd Conference on Uncertainty in Artificial Intelligence*.
[7] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference*. 689–698.
[8] Dugang Liu, Pengxiang Cheng, Zhenhua Dong, Xiuqiang He, Weike Pan, and Zhong Ming. 2020. A general knowledge distillation framework for counterfactual recommendation via uniform data. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 831–840.
[9] Dugang Liu, Chen Lin, Zhilin Zhang, Yanghua Xiao, and Hanghang Tong. 2019. Spiral of silence in recommender systems. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining*. 222–230.
[10] David C Liu, Stephanie Rogers, Raymond Shiau, Dmitry Kislyuk, Kevin C Ma, Zhigang Zhong, Jenny Liu, and Yushi Jing. 2017. Related pins at pinterest: The evolution of a real-world recommender system. In *Proceedings of the 26th International Conference on World Wide Web Companion*. 583–592.
[11] Benjamin M Marlin and Richard S Zemel. 2009. Collaborative prediction and ranking with non-random missing data. In *Proceedings of the 3rd ACM Conference on Recommender systems*. 5–12.
[12] Andriy Mnih and Ruslan R Salakhutdinov. 2007. Probabilistic matrix factorization. In *Proceedings of the 21st International Conference on Neural Information Processing Systems*. 1257–1264.
[13] Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering* 22, 10 (2010), 1345–1359.
[14] Yuta Saito, Suguru Yaginuma, Yuta Nishino, Hayato Sakata, and Kazuhide Nakata. 2020. Unbiased recommender learning from missing-not-at-random implicit feedback. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 501–509.
[15] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations as treatments: Debiasing learning and evaluation. *arXiv preprint arXiv:1602.05352* (2016).
[16] Jiangxing Yu, Hong Zhu, Chih-Yao Chang, Xinhua Feng, Bowen Yuan, Xiuqiang He, and Zhenhua Dong. 2020. Influence function for unbiased recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1929–1932.
[17] Ziwei Zhu, Yun He, Yin Zhang, and James Caverlee. 2020. Unbiased implicit recommendation and propensity estimation via combinational joint learning. In *Proceedings of the 14th ACM Conference on Recommender Systems*. 551–556.